Deep and Cooperative Learning for Approximate Dynamic Programming

MAX GREENE AND WARREN DIXON

Dept. of Mechanical and Aerospace Engineering, Univ. of Florida

Outline

- Deep Neural Network Identifier for Approximate Optimal Control
  - Online System ID using DNN
  - Update the DNN in Real-Time
  - Approximate System Dynamics
  - Use System Model to Approximate Optimal Controller

- Cooperative Approximate Optimal Control
  - Leverage Multiple Agents Learning the Same Function

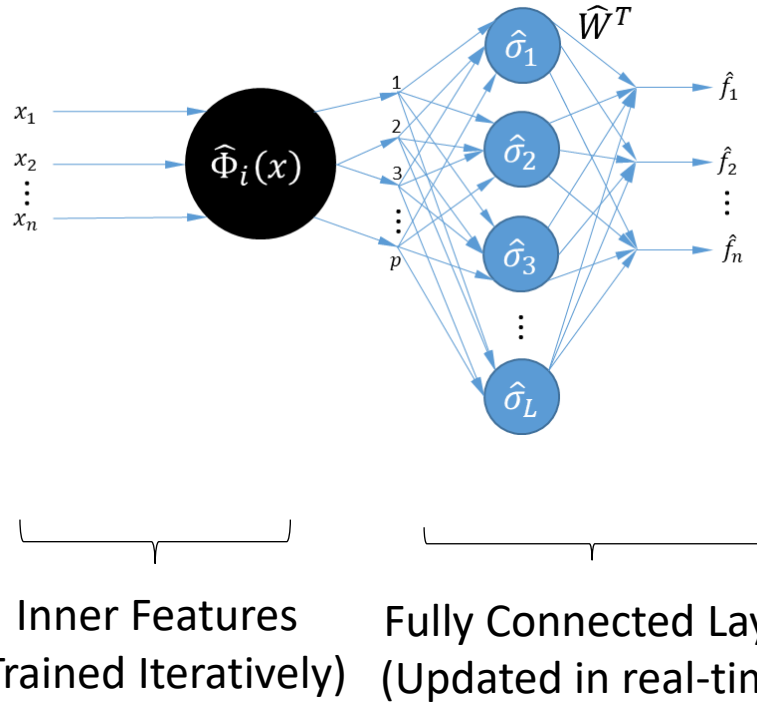
Deep Neural Network-based Approximate Optimal Tracking for Unknown Nonlinear Systems

MAX GREENE[1], ZACHARY BELL[2], SCOTT NIVISON[2], WARREN DIXON[1]

[1]Dept. of Mechanical and Aerospace Engineering, Univ. of Florida
[2]Munitions Directorate, Eglin AFB, Air Force Research Laboratory

Real-Time DNN Approximation

- Multi-Timescale DNN
  - Online system identifier
  - Better than single hidden-layer
  - Inner features updates iteratively (switched)
  - Output-layer updates in real-time
  - Can we use a DNN to approximate an optimal control policy online?



Real-Time DNN Approximation

Dynamical System

Given a control affine nonlinear dynamical system:

$$\dot{x} = f(x) + g(x)u$$

DNN Approximation

Approximate the drift dynamics with a DNN:

Dynamics: $f(x) = \theta^T \phi^*\left(\Phi^*(x)\right) + \epsilon(x)$

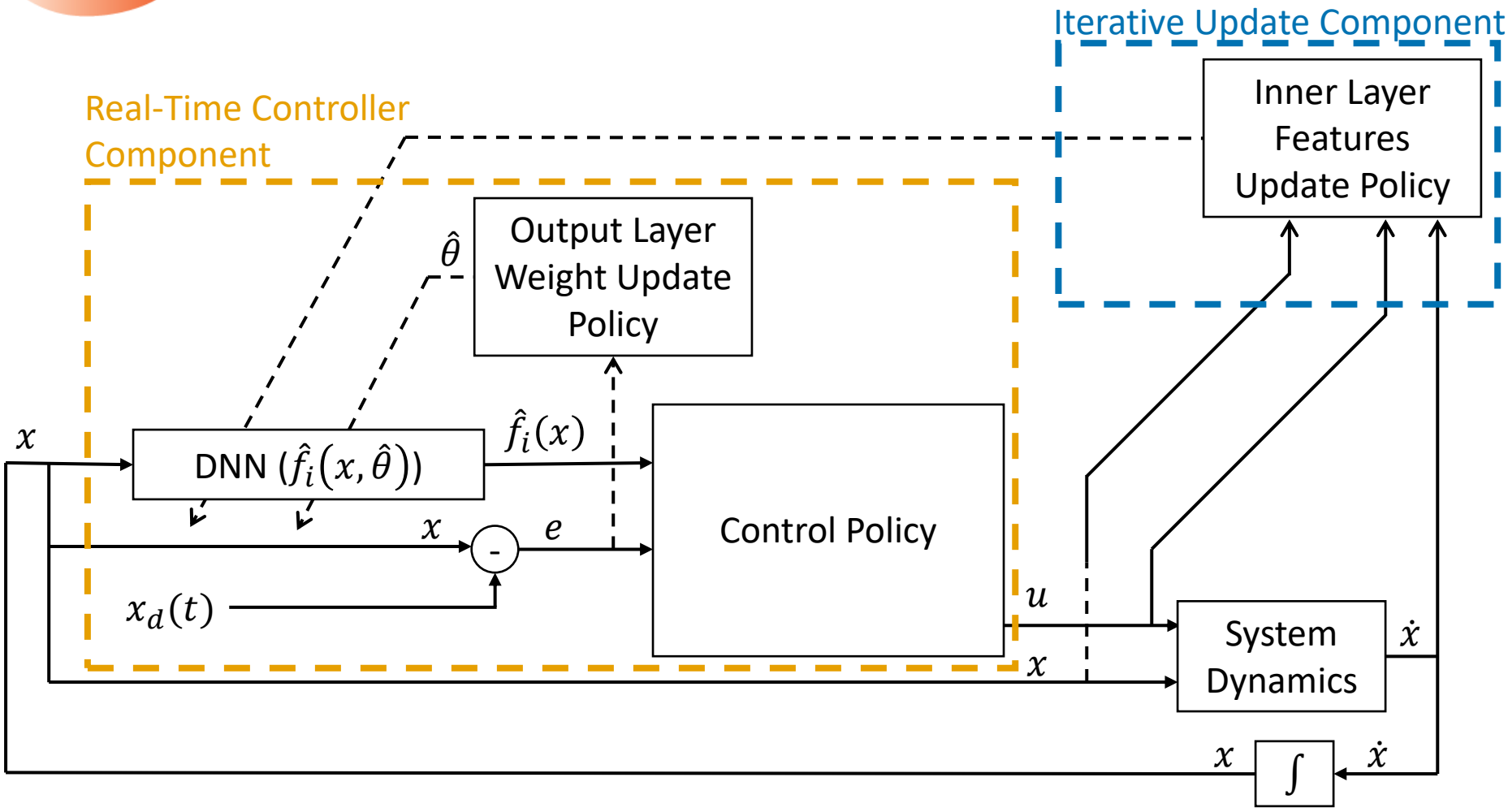
i^{th} DNN Approximation: $\hat{f}_i(x, \hat{\theta}) = \hat{\theta}^T \hat{\phi}_i\left(\hat{\Phi}_i(x)\right)$

Looks Like:

Reminds us of linearly parameterizable dynamics:

$$f(x) = \theta^T \phi^*\left(\Phi^*(x)\right) + \epsilon(x) \approx \theta^T Y(x) + \epsilon(x)$$

Real-Time DNN Approximation



UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

DNN-Identifier ADP

Dynamical System

Given a control affine nonlinear dynamical system:

$$\dot{\zeta} = F(\zeta) + G(\zeta)\mu$$

Control Objective

Design a controller, u, which minimizes a cost function:

$$J(\zeta, \mu) = \min_{u(\tau)\epsilon U} \int_0^\infty Q\big(\zeta(\tau)\big) + \mu(\tau)^T R\mu(\tau)\ d\tau$$

Cost-to-Go

Optimal value function:

$$V^*(x) = \min_{u(\tau)\epsilon U} \int_0^\infty Q\big(\zeta(\tau)\big) + \mu(\tau)^T R\mu(\tau)\ d\tau$$

Hamilton Jacobi Bellman Equation

Hamilton Jacobi Bellman (HJB) equation:

$$0 = Q(\zeta) + \mu^*(\zeta)^T R \mu^*(\zeta) + \nabla_\zeta V^*(\zeta)\big(F(\zeta) + G(\zeta)\mu^*(\zeta)\big)$$

Optimal Controller

From the HJB equation:

$$\mu^*(\zeta) = -\frac{1}{2} R^{-1} G(\zeta)^T \left(\nabla_\zeta V^*(\zeta)\right)^T$$

- Cannot solve HJB analytically.
- Approximate the Value Function (V^*)
  - Universal Function Approximation Property
  - Neural Networks

DNN-Identifier ADP

Optimal Value Function and Optimal Control Policy:

$$V^*(\zeta) = W^T\sigma(\zeta) + \varepsilon(\zeta) \qquad \mu^*(\zeta) = -\frac{1}{2}R^{-1}G(\zeta)^T\left(\nabla_\zeta\sigma(\zeta)^T W + \nabla_\zeta\varepsilon(\zeta)^T\right)$$

Unknown: Neural weights $\quad \widehat{W}_c, \widehat{W}_a \to W$

$\widehat{W}_c$: Critic weight
$\widehat{W}_a$: Actor weight

Value Function and Optimal Control Policy Approximation

$$\hat{V}\left(\zeta, \widehat{W}_c\right) = \widehat{W}_c{}^T\sigma(\zeta) \qquad \hat{\mu}\left(\zeta, \widehat{W}_a\right) = -\frac{1}{2}R^{-1}G(\zeta)^T\left(\nabla_\zeta\sigma(\zeta)^T\widehat{W}_a\right)$$

Bellman Error (BE): Residual from HJB

$$\hat{\delta}\left(\zeta, \widehat{W}_c, \widehat{W}_a\right) \triangleq Q(\zeta) + \hat{\mu}\left(\zeta, \widehat{W}_a\right)^T R\hat{\mu}\left(\zeta, \widehat{W}_a\right) + \nabla_\zeta\hat{V}\left(\zeta, \widehat{W}_c\right)\left(\hat{F}_i(\zeta) + G(\zeta)\hat{\mu}\left(\zeta, \widehat{W}_a\right)\right)$$

NN Update Laws

Instantaneous Bellman Error: Residual from HJB

$$\hat{\delta}(t) \triangleq \hat{\delta}\left(\zeta(t), \hat{W}_c(t), \hat{W}_a(t)\right)$$

Weight Update Laws using R-MBRL

$$\dot{\hat{W}}_c(t) = -\eta_{c1}\Gamma\frac{\omega(t)}{\rho(t)}\hat{\delta}(t) - \eta_{c2}\Gamma\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i(t)}{\rho_i(t)}\hat{\delta}_i(t)$$

Minimize the Bellman Error

$$\dot{\Gamma}(t) = \left(\lambda\Gamma(t) - \eta_{c1}\frac{\Gamma(t)\omega(t)\omega(t)^T\Gamma(t)}{\rho(t)} - \eta_{c2}\Gamma(t)\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i(t)\omega_i^T(t)}{\rho_i(t)}\right)\Gamma(t)\right)\mathbf{1}_{\{}$$

Adjust the "learning rate"

$$\dot{\hat{W}}_a(t) = -\eta_{c1}\left(\hat{W}_a(t) - \hat{W}_c(t)\right) - \eta_{a2}\hat{W}_a(t) + \eta_{c1}\frac{G_\sigma(t)^T\hat{W}_a(t)\omega(t)^T}{4\rho(t)}\hat{W}_c(t)$$
$$+\eta_{c2}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{G_{i\sigma}^T\hat{W}_a(t)\omega_i(t)^T}{4\rho_i(t)}\right)\hat{W}_c(t)$$

Follow the critic weights

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

NN Update Laws

Instantaneous Bellman Error: Residual from HJB

$$\hat{\delta}(t) \triangleq \hat{\delta}\left(\zeta(t), \hat{W}_c(t), \hat{W}_a(t)\right)$$

Weight Update Laws using R-MBRL

$$\dot{\hat{W}}_c(t) = -\eta_{c1}\Gamma\frac{\omega(t)}{\rho(t)}\hat{\delta}(t) - \eta_{c2}\Gamma\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i(t)}{\rho_i(t)}\hat{\delta}_i(t)$$

On-Trajectory Point

$$\dot{\Gamma}(t) = \left(\lambda\Gamma(t) - \eta_{c1}\frac{\Gamma(t)\omega(t)\omega(t)^T\Gamma(t)}{\rho(t)} - \eta_{c2}\Gamma(t)\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\omega_i(t)\omega_i^T(t)}{\rho_i(t)}\right)\Gamma(t)\right)\mathbf{1}_{\{\underline{\Gamma}\leq\|\Gamma\|\leq\bar{\Gamma}\}}$$

$$\dot{\hat{W}}_a(t) = -\eta_{c1}\left(\hat{W}_a(t) - \hat{W}_c(t)\right) - \eta_{a2}\hat{W}_a(t) + \eta_{c1}\frac{G_\sigma(t)^T\hat{W}_a(t)\omega(t)^T}{4\rho(t)}\hat{W}_c(t) + \eta_{c2}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{G_{i\sigma}^T\hat{W}_a(t)\omega_i(t)^T}{4\rho_i(t)}\right)\hat{W}_c(t)$$

Off-Trajectory Points

UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

Theorem 1

Theorem 1:

Provided the necessary assumptions, gain conditions, and initial conditions hold, then the tracking error $e(t)$, weight estimation errors $\widetilde{W}_c(t)$and $\widetilde{W}_a(t)$, system ID DNN output weight estimation errors $\tilde{\theta}(t)$, state estimator error $\tilde{x}(t)$, and transient control policy $\mu(t)$ are uniformly ultimately bounded (UUB).

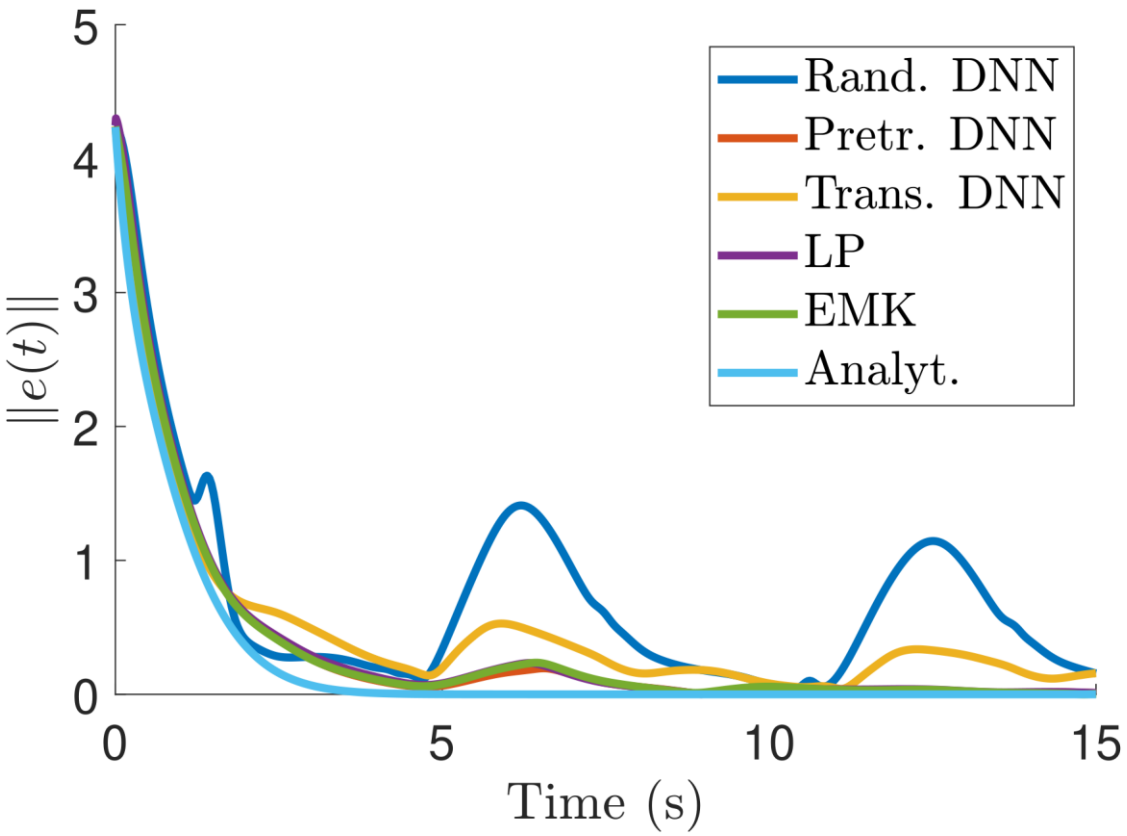
- While iteratively retraining (switching) DNN-based system ID terms:
  - $\|e\|$ converges to a neighborhood around zero
  - $\left\|\tilde{\theta}\right\|$, $\|\tilde{x}\|$ converge to a neighborhood around zero
  - $\left\|\widetilde{W}_c\right\|, \left\|\widetilde{W}_a\right\|$ converge to a neighborhood around zero
  - $\mu(t)$ is an approximation of the optimal control policy

Theorem 1

- Contribution of Theorem 1
  - Trajectory Tracking
  - Approximate Optimal Control Policy
  - Allows iteratively updating the DNN inner-layer features.

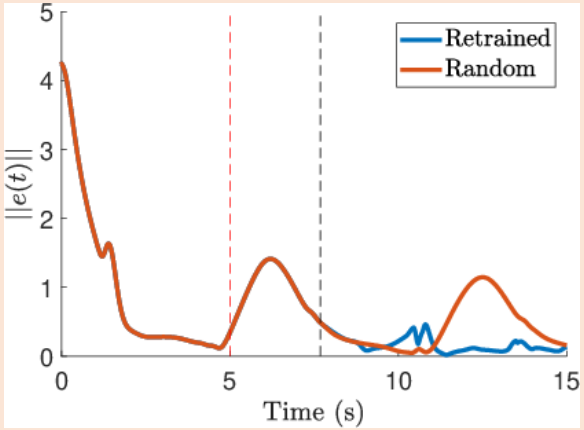
- Obstacles in Theorem 1
  - Updating the DNN introduces discontinuities
    - $\hat{\delta}(t) \triangleq \nabla_\zeta \hat{V}\left(\zeta, \widehat{W}_c\right)\left(\hat{F}_i(\zeta) + G(\zeta)\hat{\mu}\left(\zeta, \widehat{W}_a\right)\right) + \cdots$
    - $\dot{\widehat{W}}_c(t) = -\eta_{c1}\Gamma\frac{\omega(t)}{\rho(t)}\hat{\delta}(t) - \eta_{c2}\frac{1}{N}\sum_{i=1}^{N_e}\frac{\omega_i(t)}{\rho_i(t)}\hat{\delta}_i(t)$
    - $\dot{\widehat{W}}_c(t)$ is discontinuous
    - $\widehat{W}_c(t)$ is nonsmooth
    - Requires use of generalized solutions to show stability

DNN ADP Simulation Results

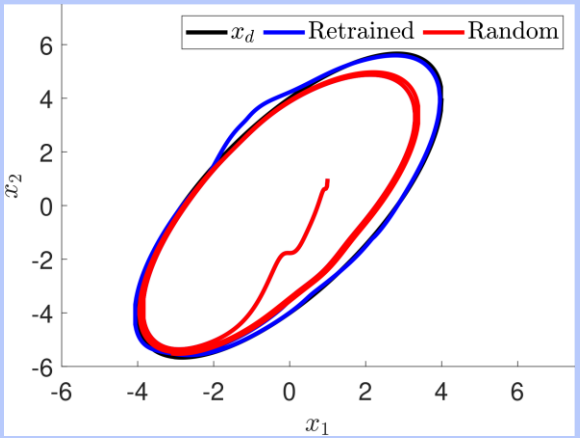


DNN ADP Simulation Results

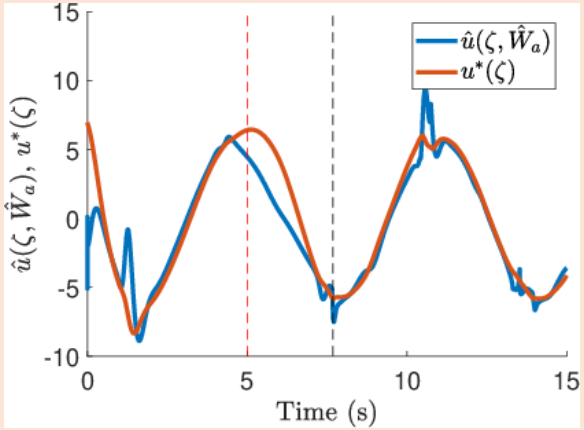
Norm of Error



Phase Plot

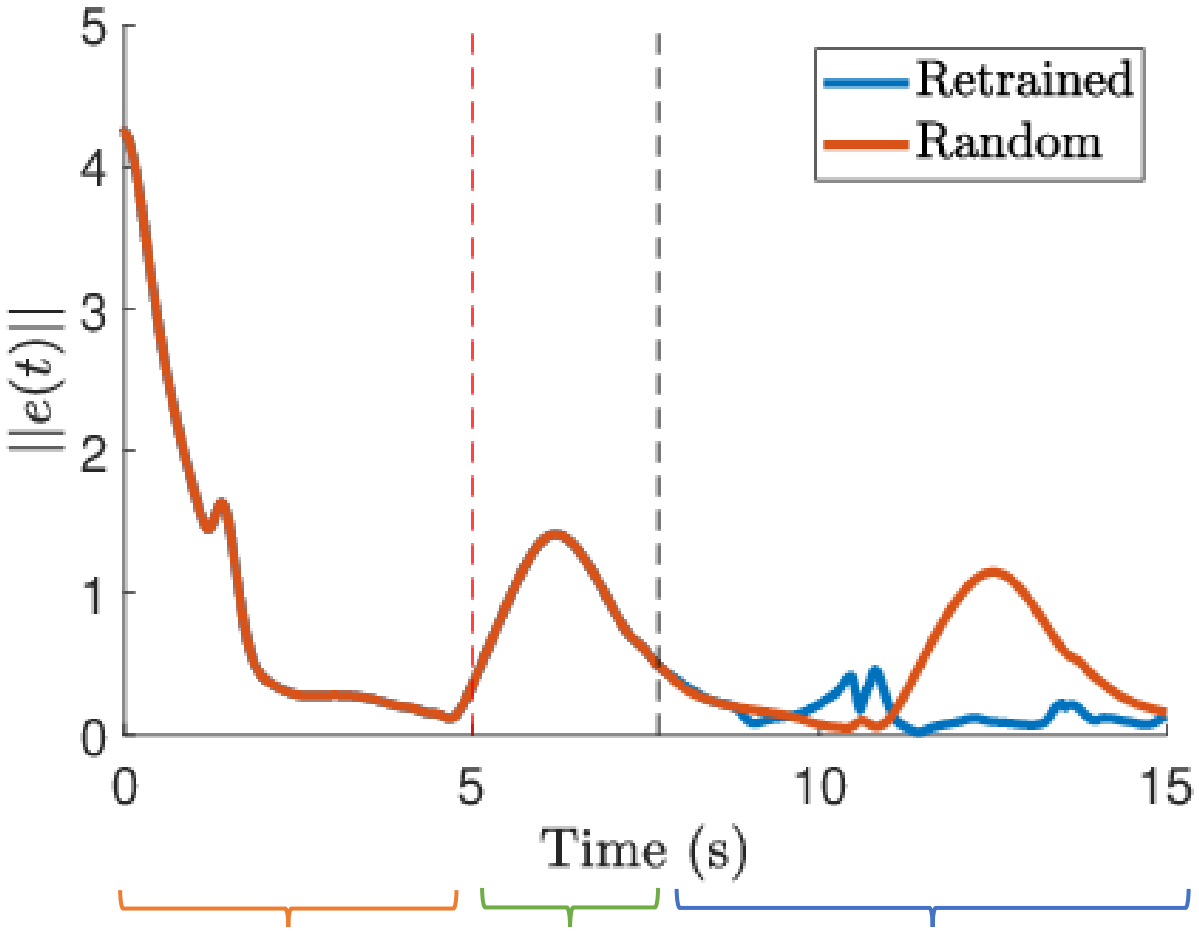


Optimal Control Comparison



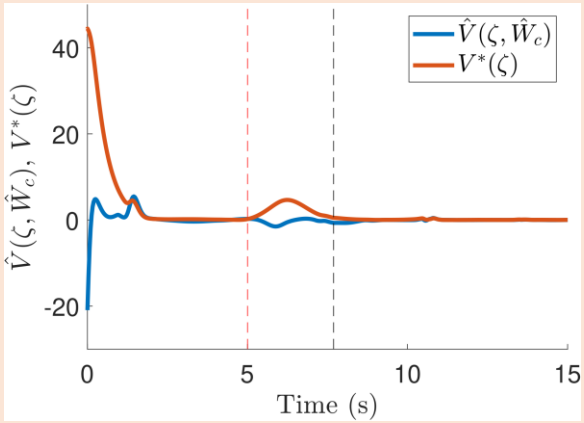
UF UNIVERSITY of FLORIDA
Duke UNIVERSITY
TEXAS The University of Texas at Austin
UC SANTA CRUZ

DNN ADP Simulation Results

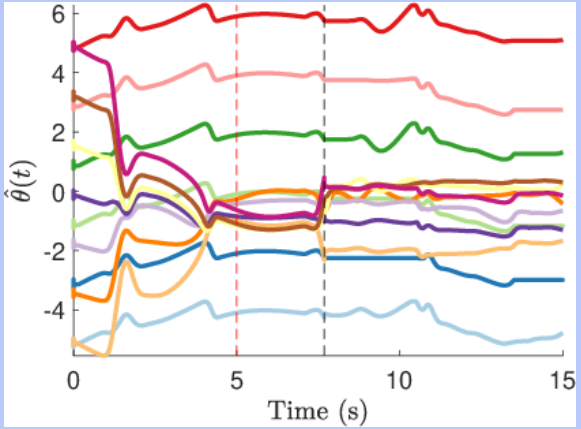


DNN ADP Simulation Results

Value Function Comparison



DNN Output Weights



UF UNIVERSITY of FLORIDA · Duke UNIVERSITY · TEXAS The University of Texas at Austin · UC SANTA CRUZ

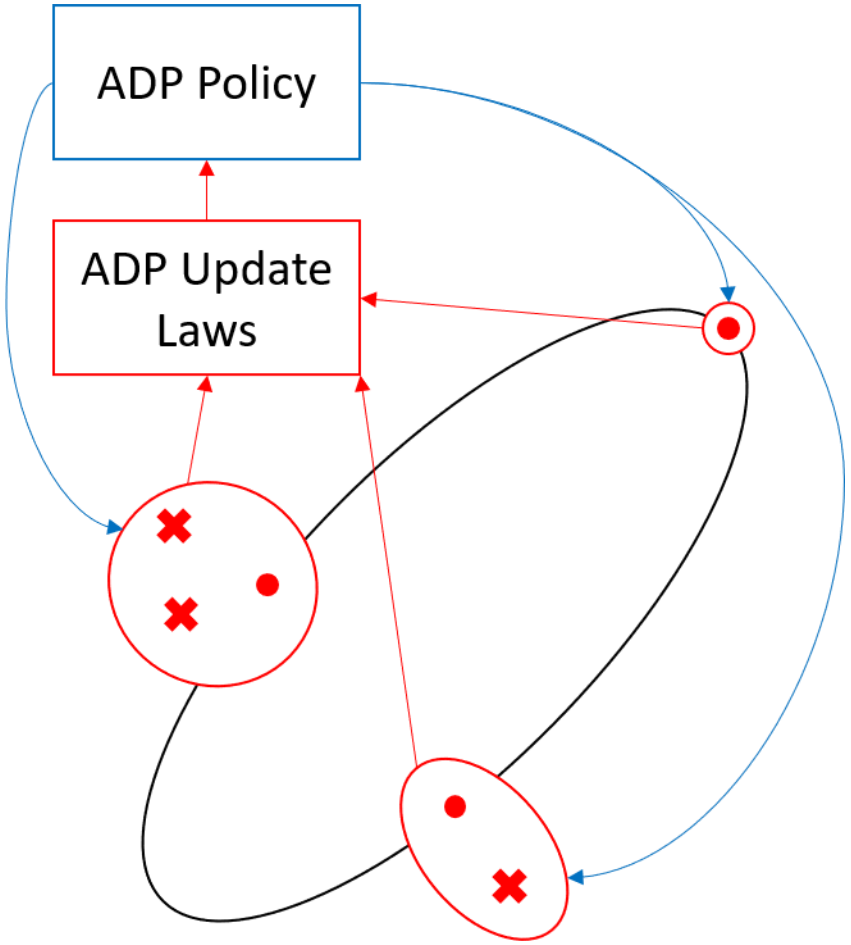
Cooperative Model-Based Reinforcement Learning for Approximate Optimal Tracking

MAX GREENE[1], ZACHARY BELL[2], SCOTT NIVISON[2], JONATHAN HOW[3], WARREN DIXON[1]

[1]Dept. of Mechanical and Aerospace Engineering, Univ. of Florida
[2]Munitions Directorate, Eglin AFB, Air Force Research Laboratory
[3]Department of Aero. and Astro., Massachusetts Institute of Technology

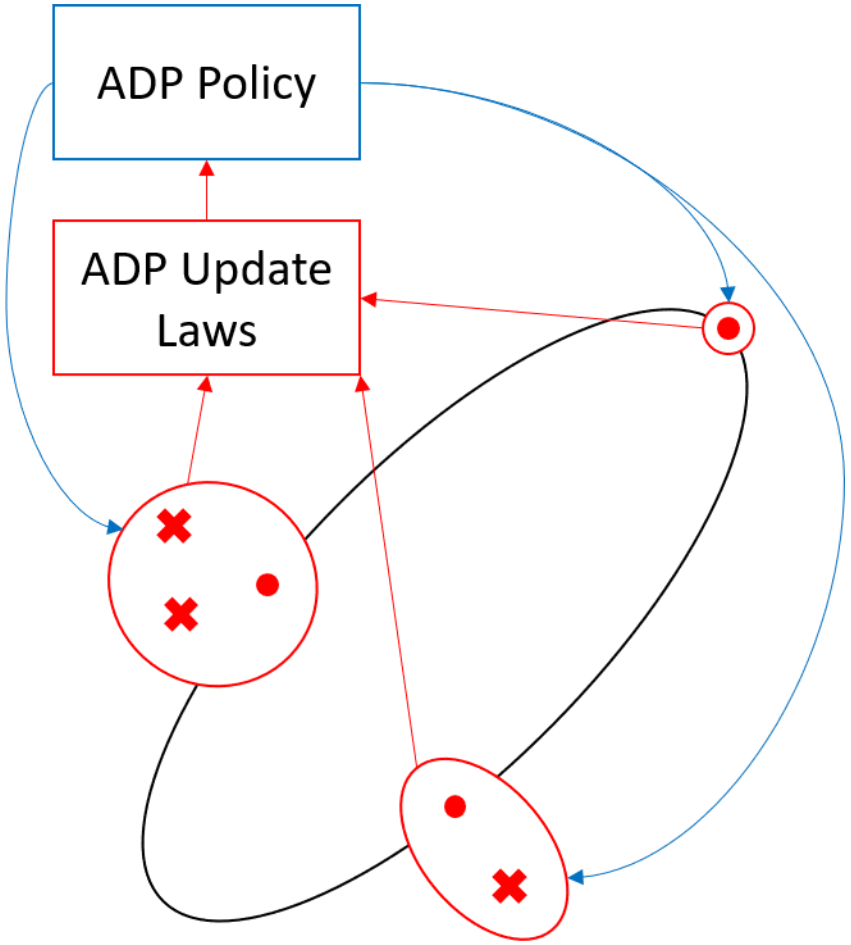
Cooperative BE Extrapolation

- Developed Method
  - Homogeneous agents
    - i.e., identical dynamics
  - Each agent computes one on- and some off-trajectory BE extrapolation points.
  - BE extrapolation is used in update laws.
  - Divides computational expense of BE extrapolation.



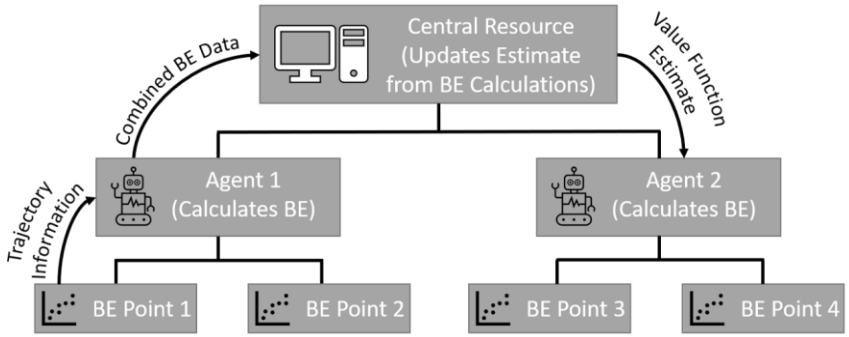
Cooperative BE Extrapolation

- Why Cooperative?
  - Share data between agents
  - Train DNN system ID
    - Collect data faster
    - More agents means more data
  - Value Function Appx.
    - Without excessive computational loads on one agent

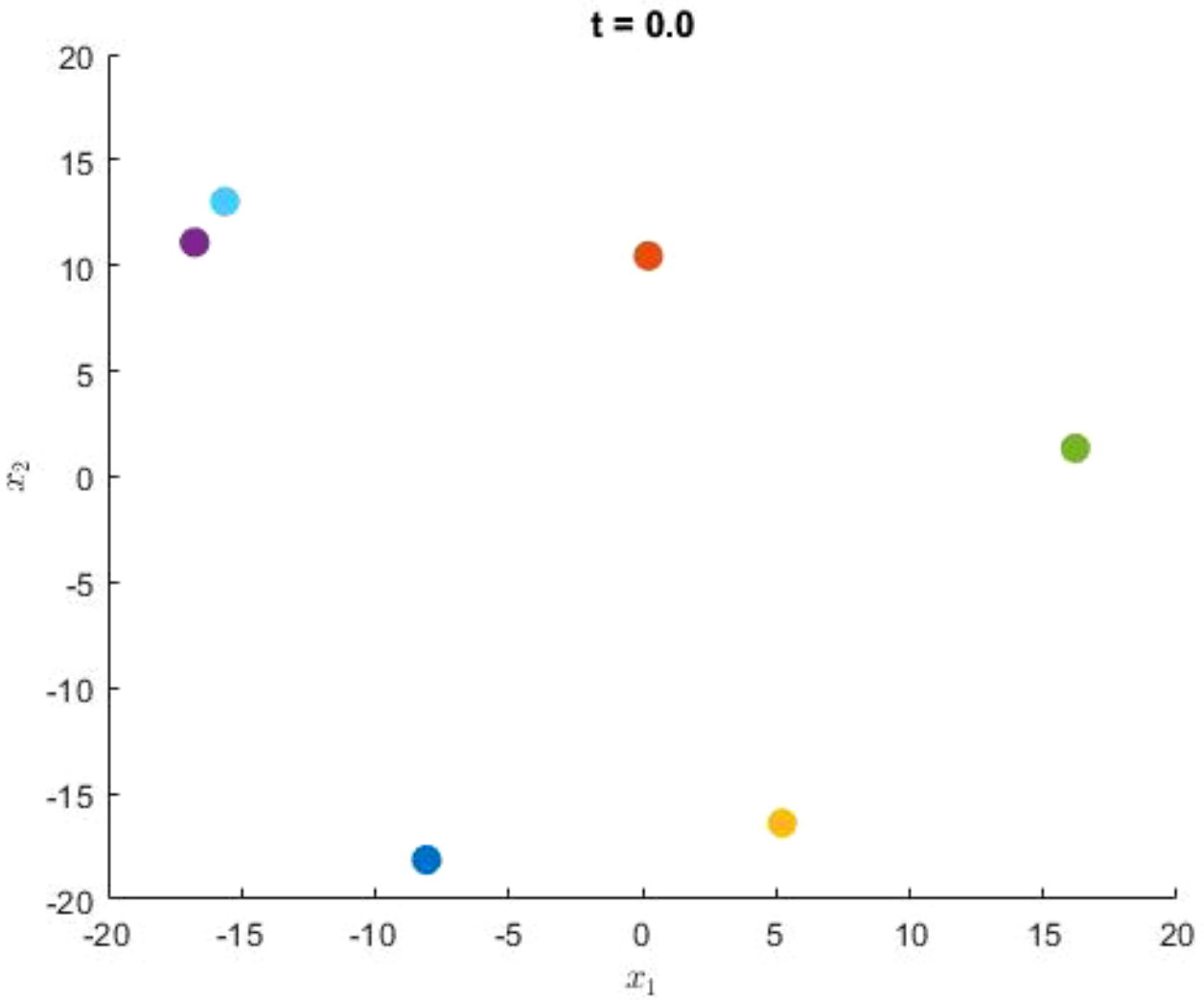


Cooperative BE Extrapolation

- Developed Method
  - Alternative view
  - Each agent calculates BE trajectory data.
  - Agent communicates BE data to central resource
  - Central resource communicates ADP policy to agents



Simulation



UF UNIVERSITY of FLORIDA

Duke UNIVERSITY

TEXAS The University of Texas at Austin

UC SANTA CRUZ

Conclusion

- Future Work
  - DNN ADP
    - Perform DNN-based value function approximation
      - Currently single-layer NN
  - Cooperative ADP
    - Extend to heterogeneous agents
    - How do we use share data between agents to cooperatively accomplish a global objective?

Questions?

MAXGREENE12@UFL.EDU